## IV VIRTUAL MEMORY

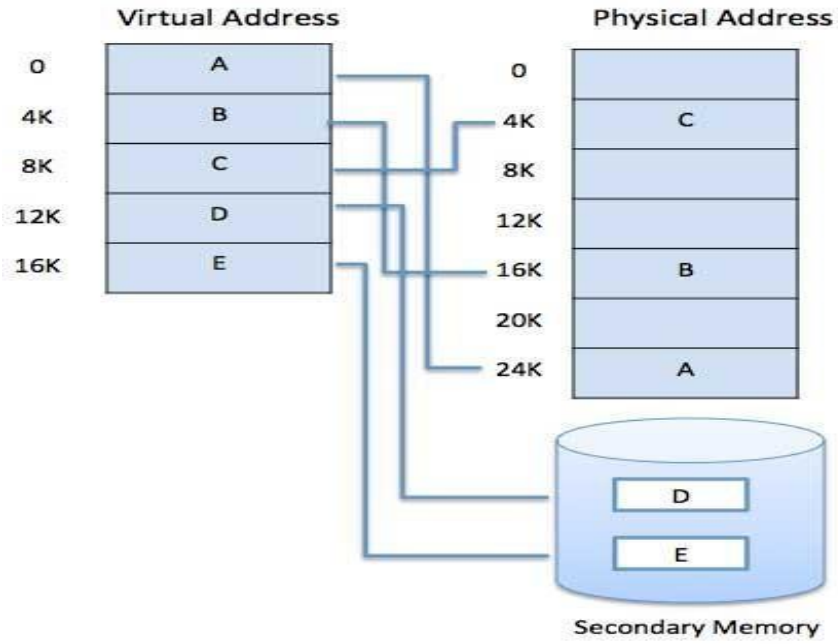
A computer can address more memory than the amount physically installed on the system. This extra memory is actually called **virtual memory** and it is a section of a hard disk that's set up to emulate the computer's RAM.

The main visible advantage of this scheme is that programs can be larger than physical memory. Virtual memory serves two purposes. First, it allows us to extend the use of physical memory by using disk. Second, it allows us to have memory protection, because each virtual address is translated to a physical address.

Following are the situations, when entire program is not required to be loaded fully in main memory.

- User written error handling routines are used only when an error occurred in the data or computation.
- Certain options and features of a program may be used rarely.
- Many tables are assigned a fixed amount of address space even though only a small amount of the tableis actually used.
- The ability to execute a program that is only partially in memory would counter many benefits.
- Less number of I/O would be needed to load or swap each user program into memory.
- A program would no longer be constrained by the amount of physical memory that is available.
- Each user program could take less physical memory, more programs could be run the same time, with acorresponding increase in CPU utilization and throughput.

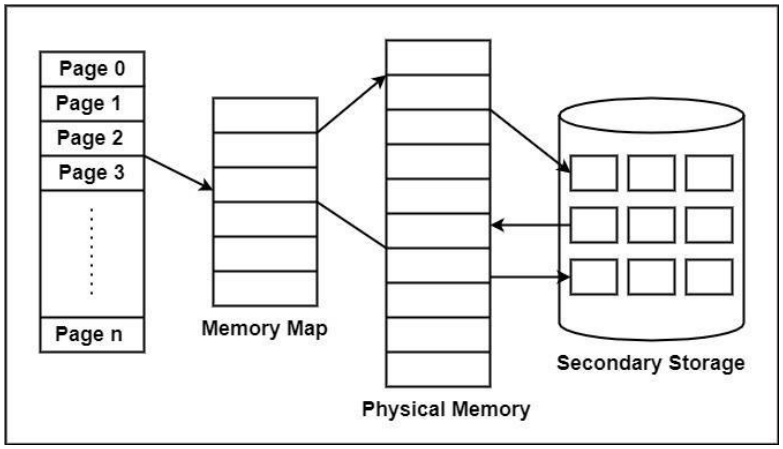
Modern microprocessors intended for general-purpose use, a memory management unit, or MMU, is built into the hardware. The MMU's job is to translate virtual addresses into physical addresses. A basic example is given below −




Virtual memory is commonly implemented by demand paging. It can also be implemented in a segmentation system. Demand segmentation can also be used to provide virtual memory.

Virtual memory is the partition of logical memory from physical memory. This partition supports large virtual memory for programmers when only limited physical memory is available.

Virtual memory can give programmers the deception that they have a very high memory although the computer has a small main memory. It creates the function of programming easier because the programmer nolonger requires to worry about the multiple physical memory available.



Virtual memory works similarly, but at one level up in the memory hierarchy. A memory management unit (MMU) transfers data between physical memory and some gradual storage device, generally a disk. This storage area can be defined as a swap disk or swap file, based on its execution. Retrieving data from physical memory is much faster than accessing data from the swap disk.

There are two primary methods for implementing virtual memory are as follows −

- *Paging*

Paging is a technique of memory management where small fixed-length pages are allocated instead of a single large variable-length contiguous block in the case of the dynamic allocation technique. In a paged system, each process is divided into several fixed-size 'chunks' called pages, typically 4k bytes in length. The memory space is also divided into blocks of the equal size known as frames.

**Advantages of Paging**

There are the following advantages of Paging are −

- In Paging, there is no requirement for external fragmentation.
- In Paging, the swapping among equal-size pages and page frames is clear.
- Paging is a simple approach that it can use for memory management.

**Disadvantage of Paging**

There are the following disadvantages of Paging are −

- In Paging, there can be a chance of Internal Fragmentation.
- In Paging, the page table employs more memory.
- Because of Multi-level Paging, there can be a chance of memory reference overhead.
- *Segmentation*

The partition of memory into logical units called segments, according to the user's perspective is called segmentation. Segmentation allows each segment to grow independently, and share. In other words, segmentation is a technique that partition memory into logically related units called a segment. It means that the program is a collection of the segment.

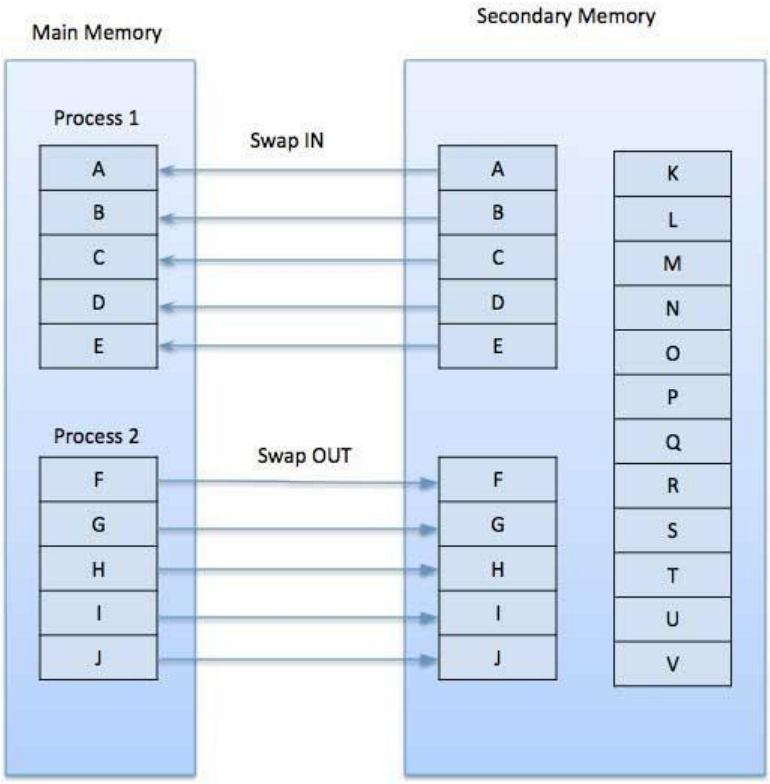
Unlike pages, segments can vary in size. This requires the MMU to manage segmented memory somewhat differently than it would manage paged memory. A segmented MMU contains a segment table to maintain track of the segments resident in memory.

A segment can initiate at one of several addresses and can be of any size, each segment table entry should

contain the start address and segment size. Some system allows a segment to start at any address, while other limits the start address. One such limit is found in the Intel X86 architecture, which requires a segment to start at an address that has 6000 as its four low-order bits.

**Demand Paging**

A demand paging system is quite similar to a paging system with swapping where processes reside in secondary memory and pages are loaded only on demand, not in advance. When a context switch occurs, the operating system does not copy any of the old program's pages out to the disk or any of the new program's pages into the main memory Instead, it just begins executing the new program after loading the first page and fetches that program's pages as they are referenced.




While executing a program, if the program references a page which is not available in the main memory because it was swapped out a little ago, the processor treats this invalid memory reference as a **page fault** and transfers control from the program to the operating system to demand the page back into the memory.

*Advantages*

Following are the advantages of Demand Paging −

- Large virtual memory.
- More efficient use of memory.
- There is no limit on degree of multiprogramming.

*Disadvantages*

- Number of tables and the amount of processor overhead for handling page interrupts are greater than inthe case of the simple paged management techniques.

Page Replacement Algorithm

Page replacement algorithms are the techniques using which an Operating System decides which memory pages to swap out, write to disk when a page of memory needs to be allocated. Paging happens whenever a page fault occurs and a free page cannot be used for allocation purpose accounting to reason that pages are not available or the number of free pages is lower than required pages.

When the page that was selected for replacement and was paged out, is referenced again, it has to read in from disk, and this requires for I/O completion. This process determines the quality of the page replacement algorithm: the lesser the time waiting for page-ins, the better is the algorithm.

A page replacement algorithm looks at the limited information about accessing the pages provided by hardware, and tries to select which pages should be replaced to minimize the total number of page misses, while balancing it with the costs of primary storage and processor time of the algorithm itself. There are many different page replacement algorithms. We evaluate an algorithm by running it on a particular string of memory reference and computing the number of page faults,

Reference String

The string of memory references is called reference string. Reference strings are generated artificially or by tracing a given system and recording the address of each memory reference. The latter choice produces a large number of data, where we note two things.

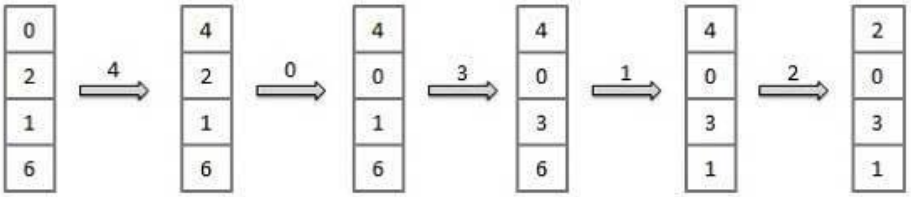
- For a given page size, we need to consider only the page number, not the entire address.
- If we have a reference to a page **p**, then any immediately following references to page **p** will never cause a page fault. Page p will be in memory after the first reference; the immediately following references will not fault.
- For example, consider the following sequence of addresses − 123,215,600,1234,76,96

- If page size is 100, then the reference string is 1,2,6,12,0,0First In First Out (FIFO) algorithm

- Oldest page in main memory is the one which will be selected for replacement.
- Easy to implement, keep a list, replace pages from the tail and add new pages at the head.

Reference String : 0, 2, 1, 6, 4, 0, 1, 0, 3, 1, 2, 1

Misses : x x x x x x x x x



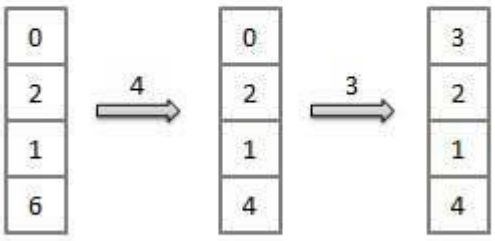

Fault Rate = 9 / 12 = 0.75

Optimal Page algorithm

- An optimal page-replacement algorithm has the lowest page-fault rate of all algorithms. An optimal page-replacement algorithm exists, and has been called OPT or MIN.
- Replace the page that will not be used for the longest period of time. Use the time when a page is to be used

Reference String : 0, 2, 1, 6, 4, 0, 1, 0, 3, 1, 2, 1

Misses : x x x x x x




Fault Rate = 6 / 12 = 0.50

Least Recently Used (LRU) algorithm

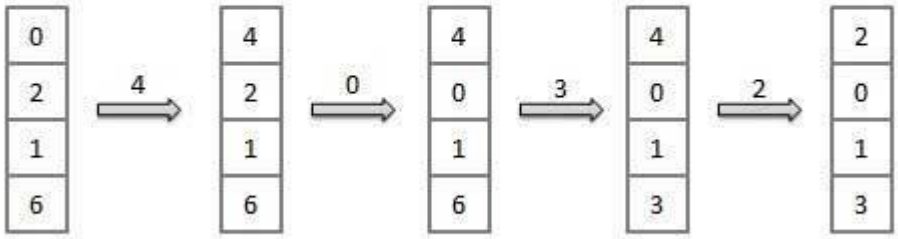
- Page which has not been used for the longest time in main memory is the one which will be selected for

replacement.

- Easy to implement, keep a list, replace pages by looking back into time.

Reference String : 0, 2, 1, 6, 4, 0, 1, 0, 3, 1, 2, 1

Misses : x x x x x x x x




Fault Rate = 8 / 12 = 0.67

Page Buffering algorithm

- To get a process start quickly, keep a pool of free frames.
- On page fault, select a page to be replaced.
- Write the new page in the frame of free pool, mark the page table and restart the process.
- Now write the dirty page out of disk and place the frame holding replaced page in free pool.

Least frequently Used(LFU) algorithm

- The page with the smallest count is the one which will be selected for replacement.
- This algorithm suffers from the situation in which a page is used heavily during the initial phase of a process, but then is never used again.

**Most frequently Used(MFU) algorithm**

- This algorithm is based on the argument that the page with the smallest count was probably just broughtin and has yet to be used.